|  |  |
| --- | --- |
| T |  |
| Attendees |  |
| Absent |  |

Introspect Site

# Introduction

Introspect is a free, [open-source](http://en.wikipedia.org/wiki/Open-source_software), light weight [application framework](http://en.wikipedia.org/wiki/Application_framework). It provides a standardized structure and functionality for developing a wide range of Java applications.

Introspect is based on the [Naked Object Design Pattern](http://en.wikipedia.org/wiki/Naked_objects).

This means that you develop your domain objects and Introspect does the rest. As a consequence:

* you only focus on the [domain](http://en.wikipedia.org/wiki/Domain_layer#Business_logic_layer)
* you can prototype, develop and maintain apps very quick and efficient. The framework will completely eliminate the need to develop/ maintain user interface code.
* you don’t put business logic outside the domain

Introspect is modular and extensible

* Introspect is [modular designed](http://en.wikipedia.org/wiki/Modular_programming). It has different **modules** that each have a specific function
  + user interface
  + authorization
  + language
  + validation
  + data access
  + reporting
* Each module has a simple interface so that each functionality can have multiple implementations. In example; Introspect has multiple user interface implementations for a:
  + Command line application
  + Desktop application (based on Swing)
  + Web application (based on the Vaadin)
  + Mobile applications (based on Android)
* You can write your own implementation or modify or extend an existing implementation

Introspect is simple

The introspect framework is simple to learn. Being a Java expert or not, you can write your applications in minutes:

* No setting up your **IDE**: You can use any java IDE. No need to install plug-ins or tools
* No setting up configuration files. The frame work initialization is simple and done programmatically. Introspect favours convention over configuration. Annotations are used when the additional information is required.
* See the **getting started videos**

Introspect is a light weight application framework

* Introspect contains only few classes and is only a few kilobytes
* The interfaces of the Introspect modules are small and simple
* Introspect has a minimal impact on an application. That is, it does not require many code changes to incorporate them into your application. You do not have to think too much about the underlying framework because there really isn't much code to write that explicitly ties you in with the "plumbing". On the other hand, traditional J2EE development with EJB entails writing a lot of "plumbing" code which weighs you down conceptually.

# Naked Objects Design Pattern

The UI representations  
correspond directly  
with the underlying  
domain object model

So, for instance:

objects instances exposed as icons

object properties / collections exposed in forms

object methods exposed as menu items

eg Claim#submit(Approver)

repositories/domain services exposed as desktop icons

eg ClaimRepository, EmployeeRepository

The “don’t repeat yourself” (DRY) principle is the idea that every concept, business rule and validation should be expressed in one place and one place only. The term was originally coined by Dave Thomas and Andy Hunt in their book, the Pragmatic Programmer (2001), and has been widely quoted since then as sound advice.

Object/relational mappers such as Hibernate are a good example of the DRY principle; rather than writing lots of boilerplate JDBC to insert, update and delete objects into the database, we instead defining a mapping and let the ORM do the heavy lifting for you.

The naked objects pattern is another example of DRY, but this time applied to the presentation layer rather than the persistence layer. So, object instances are automatically exposed as icons, while the object can be opened up into forms showing the object’s state (properties and collections). Furthermore all other public the object’s methods (we call them actions) are rendered as menu items or links.

It’s worth contrasting this with other tools that can generate CRUD applications. First, the UI is generated at runtime, not compile-time (there’s no “generate scaffolding” command to run). Second, exposing object actions means this is more than just simple CRUD style applications. Third, with Apache Isis we always starts with the domain layer. Some other tools start with either a database schema (ie reverse engineering a data model), or start with the presentation layer (where the domain model can end up as a 2nd class citizen as the domain expert gets distracted by UI concerns).

# The Introspect Architecture

# Yet another application framework

Why develop yet another application framework? There are so many available today. The answers:

1. Because its fun. Developing this framework is both challenging and educative
2. Because I love the idea of Naked Objects and Domain Driven Design but (cocky as I am) had some issues with the Apache Isis frame work:
   1. I no longer consider it to be lightweight
   2. I do not like my domain objects or domain service classes to extend something
   3. I do not like a framework to “depend” on a build tool like Maven
   4. I do not like the way the viewers looked

# 